PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-312109
(43)Date of publication of application : 09.11.1999

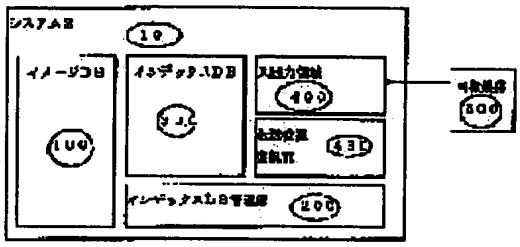
(51)Int.Cl. G06F 12/00
G06F 12/00

(21)Application number : 10-118125
(22)Date of filing : 28.04.1998

(71)Applicant : HITACHI LTD
(72)Inventor : MURAKAMI TATSUYA
UCHIYAMA TAKASHI

(54) RELATIONAL DATA BASE SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To readily move a part of a data base(DB) from a system for managing files by using a relational data base(RDB) to another system by providing the system with an I/O area for DB files and storing data of volume corresponding to a single carriable piece medium in the I/O area so as to always segment the data.
SOLUTION: When the part of a DB in a data management system using the RDB to another system is transferred, the partial DB separated and copied from the original DB cannot function as a DB. As a result when the data of a large scale system is transferred by using a carriable medium such as an MO, index data for the whole system are required to be transferred through a conventional method. Thereby the system is provided with the I/O area for inputting/outputting a data file to/from the RDB, and the DB file is transferred between systems by using the I/O area. Data files managed by the DB are recorded in the same carriable medium.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's

decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平11－312109

(43)公開日　平成11年(1999)11月９日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　12/00	５２０	Ｇ０６Ｆ　12/00	５２０Ａ
	５１２		５１２

審査請求　未請求　請求項の数６　ＯＬ　（全６頁）

(21)出願番号　特願平10－118125

(22)出願日　平成10年(1998)４月28日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　村上　達也
神奈川県小田原市国府津2880番地　株式会社日立製作所ストレージシステム事業部内

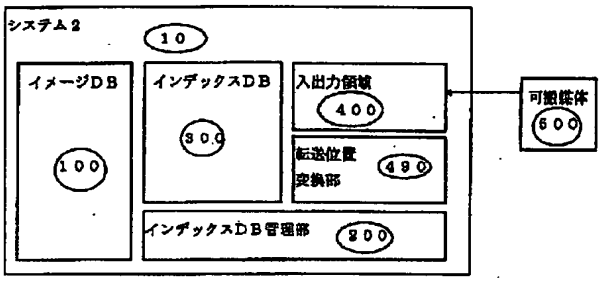
(72)発明者　内山　隆司
神奈川県小田原市国府津2880番地　株式会社日立製作所ストレージシステム事業部内

(74)代理人　弁理士　小川　勝男

(54)【発明の名称】　リレーショナルデータベースシステム

(57)【要約】
【課題】リレーショナルデータベース（ＲＤＢ）を用いてファイル管理を行うシステムにおいて、ＤＢのファイルとして入出力専用の領域を設け、可搬媒体１枚分に相当する量のデータを常に切り出せる形で保管することにより、ＤＢの一部を容易に別のシステムに移動できる。
【解決手段】ＲＤＢによるデータ管理システムでは、ＤＢの一部を別のシステムに移動する場合に、単にＤＢの一部を切り離してコピーするだけではＤＢとして機能できない。このため大規模システムのデータをＭＯ等の可搬媒体を用いて移動するには、インデックスデータはシステム全体分を移動しなければならなかった。ＲＤＢのデータファイルとして入出力専用の領域を設け、その領域を用いてシステム内外のＤＢファイルの受け渡しを行う。ＤＢにより管理されたデータファイルは、同一の可搬媒体に記録する。

図１



【特許請求の範囲】
【請求項１】データベースの一部を可搬媒体へ複製する機能を有し、該可搬媒体に複製されたデータベースの一部を、データベースとして利用できる機能を有するリレーショナルデータベースシステム。
【請求項２】リレーショナルデータベースシステムにおいて、各データを管理するデータベース制御部と、管理するデータを記録するデータベースファイルと、該データの一部のみからなる入出力用のデータベースファイルと、可搬媒体へのデータ入出力手段と、該入出力用データベースの内容を該可搬媒体へ移動する手段を持つことを特徴とするリレーショナルデータベースシステム。
【請求項３】リレーショナルデータベースシステムにおいて、管理しているデータベースの一部を登録する入出力用データベースを持つことを特徴とするリレーショナルデータベースシステム。
【請求項４】請求項２のデータベースシステムにおいて、データを登録する場合にデータベースファイルと入出力用データベースファイルに同時に記録する手段を持つことを特徴とするリレーショナルデータベースシステム。
【請求項５】リレーショナルデータベースシステムにおいて、各データを管理するデータベース制御部と、管理するデータを記録する複数のデータベースファイルと、登録されるデータに応じて登録先のデータベースを切り替える手段と、各データベースの内容をそれぞれ別の可搬媒体に移動する手段を有することを特徴とするデータベースシステム。
【請求項６】リレーショナルデータベースシステムにおいて、各データを管理するデータベース制御部と、管理するデータを記録する複数のデータベースファイルと、複数の可搬媒体から該複数のデータベースファイルにデータを移動する手段と、該複数のデータベースを間を同一のユーザによりアクセスする手段を有することを特徴とするデータベースシステム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、電子計算機において多量のデータを管理するデータベースシステムに関し、特に、大量のデータファイルをデータベースを用いて管理するシステムにおいて、内部に蓄積された大規模なデータの一部を、可搬媒体上に複製して別のシステムに移動して利用することを可能とするデータ管理方法に関するものである。
【０００２】
【従来の技術】計算機システムにおいて、大量のデータファイルを管理する場合、各データ毎にインデックスを付け、そのインデックスを用いて管理又は検索を行う方法が一般的である。
【０００３】ここで、インデックスの管理には大別して２種類の方法がある。一つは、各項目を予め定められた１つの表に蓄積し、各行単位で関連を持たせるテーブルにより管理する方法である。残る一つは、２項目のテーブルを複数組み合わせることで複雑な情報を記録するリレーショナルデータベースを用いる方法である。テーブルによりインデックスを管理した場合に、登録されているファイルの一部を切り出す際には、インデックスも該当する行の部分だけを切り出すことができる。一方、リレーショナルデータベースによりインデックスを管理した場合は、システムの構築後でもデータ構造を任意に変更することが容易である。
【０００４】
【発明が解決しようとする課題】テーブル形式でインデックスを保存した場合には、可搬媒体上にファイルとそれに対応する部分のインデックスを書き込めば、容易に、単一媒体に対するデータの移動が行える。しかし、この方法では、複数媒体にまたがるような大規模なシステムで、各媒体のインデックスを一度に検索するような場合には、一旦、計算機システムに存在するワークファイル上で、全てのインデックス情報を合成する必要がある。このとき各媒体のテーブルは単に連続しているだけであり、規模の増加に伴い、検索には多大な時間を要することになる。
【０００５】内容をソートした形で検索するためには、個々のインデックスを合成し、改めてソートしなおすような処理が必要である。また複数の媒体間でデータ構造が異なる場合は、それらを一括して管理することができない。
【０００６】一方、リレーショナルデータベース（以後、ＲＤＢと略す。）では、複数のテーブルをたどりながら目的のデータを取得することになる。そのため、あるデータをたどるときには各テーブルで、そのデータに関連する情報の位置は異なったものとなる。したがって、データベース（以後、ＤＢと略す。）上の一部のデータを切り出して、別のシステムに移動させようとした場合に、各テーブルから同一の位置に存在する情報の一部分だけを切り出しても、ＤＢとしては機能できない。
【０００７】イメージなどの容量の大きなファイルの管理にＲＤＢを用いると、そのファイルの一部、例えば、イメージのデータ部分は、光磁気ディスク媒体（以後、ＭＯと略す。）などにより容易に移動できるが、そのイメージに関連するインデックス情報は、ＲＤＢ全体を１つの単位として移動しなければならない。このためイメージはＭＯに蓄積しながら、そのインデックスは、ＤＡＴなどの大容量記憶媒体を用いて保存しなければならない事例が生じる。
【０００８】また、ＲＤＢの容量が複数の媒体を必要とする場合には、そのうち一部の媒体だけではＲＤＢの内容を復元することは不可能であり、必ず全ての媒体をまとめて移動する必要がある。この結果、既に多量のデー

タファイルとそれを管理する大規模なインデックスＤＢを有するシステムに、外部から新たなデータを可搬媒体で追加する際に、インデックスデータを各データ毎に辿り、既存のＤＢへ個々に追加登録する、インポート処理が必要となる。

【０００９】

【課題を解決するための手段】本発明では、上記の課題を解決するため、インデックスの管理にＲＤＢを用いる。そしてインデックスの一部をデータファイルと共に移動するため、ＲＤＢに付属して小規模な入出力領域を予め用意する。また、この入出力領域に対してＤＢの管理プログラムには、大規模ＤＢと同じ条件で接続する機能を持たせる。具体的には、同一のユーザ名を登録し、各ユーザに同一権限を与える等である。他のシステムより可搬媒体で移動してきたファイルと、ＲＤＢ内部の情報（データ部分）は、そのまま上記の入出力領域にコピー（複製）する手段を持つ。

【００１０】この結果、ここで入力されたファイルのインデックスは小規模な独立したＲＤＢとして扱うことにより、既にユーザが有する大規模なＤＢへインポート処理を行わずとも媒体内部のデータを確認することが可能となる。

【００１１】

【発明の実施の形態】本発明の利用分野として最も効果的なイメージ管理システムを例に、より詳しくは、ＲＤＢによるインデックス管理を用いた大規模なイメージ管理システムにおいて、そのデータの一部をＭＯなどの記憶媒体により移動することを例に、以下、図を用いて本発明の実施の形態の一例を説明する。イメージ管理システムでは、各イメージは個別のファイルとして扱われる。したがって、当該ファイルを可搬媒体にコピーすることにより、容易にシステム間の移動ができることが大きなメリットとなっている。そのため、既存のイメージ管理システムの多くは、イメージファイルと共に、そのファイルに対応したインデックスデータを同じ記憶媒体であるＭＯなどの光媒体に蓄積していた。別のシステム（装置）にデータを移動する場合には、ＭＯ１枚を運ぶだけで、そのままデータを利用できるという機能を備えていた。

【００１２】これに対して、近年提案されているシステムでは、インデックス情報をＲＤＢで管理することにより汎用性及び検索機能の高いシステムを構築することを目的としている。

【００１３】図１に本発明の全体図を示す。なお、本図では各機能をブロック化して表現しているが、実際には、ソフトウェアにて機能を実現することも可能である。１００は、多数のイメージデータファイルを蓄積するイメージＤＢ、２００がデータベースの管理部分、３００が該イメージデータのインデックス情報を管理するＲＤＢのデータ部分、４００がＲＤＢの一部を可搬媒体５００に移動する際に用いる入出力領域のデータ部分、１０がシステム全体を搭載したサーバを示す。

【００１４】ここで、このシステムに蓄積されているイメージデータの総数を１００万件とし、各イメージデータの量を１００ＫＢ、１件当たりに付けられるインデックスデータを１０ＫＢとすると、このシステムに搭載されるイメージデータの総量は１００ＧＢ、インデックスデータは１０ＧＢになり、データ１件当たりの蓄積に要するデータ量は１１０ＫＢとなる。

【００１５】本システムに接続する可搬媒体５００の容量を２ＧＢとすると、１枚の媒体に記録できるデータ数は約１８０００件になる。

【００１６】入出力領域３００はＲＤＢである。その容量は、２ＧＢの容量を有する可搬媒体５００に搭載できる１８０００件分のインデックスを蓄積できる容量が必要であり、約２００ＭＢとなる。

【００１７】はじめに別のシステム（システム１）で作成されたデータを可搬媒体５００に登録し、本システム（システム２）に搬入した場合について説明する。図１で、１０がシステム２にあたる。システム１で登録されたデータを可搬媒体５００によりシステム２（１０）に装填した場合、媒体中のインデックス領域を入出力領域４００にコピーする。

【００１８】図２にインデックスの内部のテーブルの構造の一例を示す。３１０が検索で用いるイメージファイルのタイトルと各ファイルに付けられたイメージＩＤを記録するタイトルテーブル、３２０が各ＩＤとイメージファイルの名称を記録するファイル名称テーブルである。一方、ＤＢはシステム自体の情報を管理するテーブルも持つ。３５０は本システムを利用する各ユーザを管理するためユーザ名とユーザのＩＤを記録するユーザテーブルである。

【００１９】各イメージファイルは各ユーザ毎にアクセスの可否を設定する。３３０は各イメージ毎にアクセスを許すユーザのＩＤを記録したセキュリティテーブルである。

【００２０】各システムは自身へのアクセス権限を持つユーザをユーザテーブルにあらかじめ登録しておく。本実施の態様では、２人のユーザ、“User1”と“User2“を例に説明する。ここで、テーブル３５０では“ユーザ1”にＩＤ“AAA”、“ユーザ2”にＩＤ“BBB”を関連付ける。

【００２１】一方、イメージデータファイルの登録では、“文書１”のタイトルで登録する場合を例に各インデックスの内容を説明する。まず、タイトルテーブル３１０に、タイトル“文書１”とファイルのＩＤ“００１”を記録する。この“文書１”を示すイメージデータのファイル名称“img1.xxx”をテーブル３２０に、イメージＩＤ“００１”と共に記録する。各文書毎にどのユーザに対してアクセスを許可するかを記すのが、セキュ

リティテーブル３３０である。この例では“文書１”つまりイメージＩＤ“００１”が“User1”に対してのみアクセスを許可し、“文書２”つまりイメージＩＤ“００２”に対しては、“User1”および“User2”共に許可することを示す。セキュリティテーブル３３０には、各イメージＩＤとそのイメージへのアクセスを許可するユーザのＩＤが記録されている。ここでは説明を簡単にするため、セキュリティのレベルは“許可”か“否”かの１つだけであるが、実際には公知の方法により複数のレベルを設定できる。

【００２２】データの移動を行う場合、システム２はシステム１が登録しているユーザを全て登録してある必要がある。つまり、システム１のユーザテーブル３５０とセキュリティテーブル３３０の内容にはシステム２のユーザテーブルとセキュリティテーブルの内容を全て含んでいることが必要である。

【００２３】システム２は、通常は、ＲＤＢの管理部２００がＲＤＢデータ３００に接続して動作し、イメージファイル１００へのアクセスを制御する。システム１から可搬媒体５００によりシステム２にデータを移動する場合には、データのうち上記各インデックスデータは入出力領域４００にコピーされ、ＲＤＢ管理部２００はデータ３００に代わり入出力領域４００に接続して動作することで、一時的に可搬媒体５００中のイメージデータにアクセスできるようになる。

【００２４】このようにすればシステム２が、入出力領域４００を介して、可搬媒体５００に存在するインデックスデータにアクセスできるので、システム２のインデックスＤＢ３００に対して、別途、可搬媒体５００からシステム２に対するインポート処理を施す必要がない。このため短時間でシステム１のＤＢ内容（可搬媒体５００）の確認が可能となる。

【００２５】次に、図３を用いて転送位置変換部４９０を説明する。システム２が、ＲＤＢデータであると認識して、入出力領域４００の内容をアクセスしている場合には、イメージデータの記録先は可搬媒体５００の内部にある。システム１でイメージを登録した時点では、イメージデータはシステム１内に存在するため、そのままではイメージデータとの連携がとれない。

【００２６】そのため、ＩＤとイメージファイルの位置情報を記録したテーブル３２０において、変換を行う必要がある。しかしながら個々のデータを変換しながら入出力領域４００に記録するのでは、短時間の立ち上げが不可能となる。したがってサーバ１０から入出力領域４００へアクセスするときは、イメージファイルパスは先頭に可搬媒体の位置を自動で添付して応答するように、入出力領域４００とＤＢ管理部３００の間に転送位置変換部４９０を有する。ここでは、イメージファイルのパス名を返す場合に、パスの先頭に可搬媒体５００のパスを付け加えて送るものである。

【００２７】転送位置変換部４９０（図３）の動作は次の通り。まず登録元のシステム１においてイメージデータimg1.xxxの位置が￥ＩＭＡＧＥ￥ＦＯＬＤＥＲ￥であったとする（３２０、図２）。このとき各イメージのパス名としては、￥ＩＭＡＧＥ￥ＦＯＬＤＥＲに連続して、例えば￥、ＩＭＡＧＥ￥ＦＯＬＤＥＲ￥ｉｍｇ１．ｘｘｘとしてテーブル３２０に記録される。これを可搬媒体５００にコピーしシステム２に移動した場合には、イメージデータの位置はシステム２の￥ＩＭＡＧＥ￥ＦＯＬＤＥＲになければならない。

【００２８】しかしながらシステム２ではシステム２自身のファイル体系が存在する。またシステム２への可搬媒体５００からのデータの移動は、インデックスデータのみをコピーするものであり（本願発明）、イメージファイルは可搬媒体５００上に残っている。したがって、可搬媒体５００への移動（コピー）を前提としたインデックス情報によれば、システム１又はシステム２自身のファイル体系に関係無く、常に、可搬媒体５００上にイメージデータが存在する形でインデックスを付けることにより上記の問題を解決できる。

【００２９】具体的には、インデックスＲＤＢのデータ部分、インデックスＤＢ３００の中では、ファイル位置テーブル３２０の内容は、“IMAGE1”と￥ＩＭＡＧＥ￥ＦＯＬＤＥＲ￥ｉｍｇ１．ｘｘｘをリンクさせて登録する。入出力領域４００内のファイル位置テーブル４２０（図３）に、このデータを記録する際には ¥MO1¥img1.xxxとして記録する。この記録の際にパスを書き換えることを転送位置変換部４９０にて実施する。

【００３０】この“MO1”は可搬媒体上にあることを示す各システム間共通のパス名であり、実際には“MO1”以下に多階層の構造を持たせることもできる。この結果、システム２において入出力領域４００にコピーされたＲＤＢのデータで、パス名のテーブル３２０では、“IMAGE1”のパス名として、￥MO1￥ＦＯＬＤＥＲ￥ｉｍｇ１．ｘｘｘが記録されることになる。

【００３１】このため入出力領域４００のデータを扱うようにＤＢを切り替えれば、可搬媒体５００上のイメージデータのパスが直接、入出力領域４００上のＤＢからアクセスできることになる。これにより可搬媒体５００により、イメージを登録したシステム１から別のシステム２へ、メージデータとそのインデックス情報とを移動する場合にインポート処理を行わずに、可搬媒体５００のデータへシステム２がアクセスすることが可能となる。

【００３２】次に、システム２においてイメージを登録する際の動作について説明する。イメージを登録する場合、イメージデータをイメージＤＢ１００に記録し、そのインデックスデータをインデックスＤＢ３００に記録する。しかしながら、該データを可搬媒体５００を用いて別のシステムに移動することを前提とする場合は、同

時に入出力領域４００にもインデックスを記録する。このとき前述の通り、ユーザ名、セキュリティ情報等の各システムに依存するデータは、両方（システム２及び可搬媒体５００）に同一のものを予め設定しておく。

【００３３】通常のデータベース管理部であるインデックスＤＢ３００は、１つのオペレーションは１つのデータベースに対してのみ実行する。本願発明では、これを複数のＤＢファイルに同時に実施する機能を持たせることにより本機能を実現している。

【００３４】図３を用いてデータ登録の際の各ＤＢとそのテーブルへの処理を説明する。図３の４１０、４２０、４３０、４５０は、それぞれ、入出力領域４００におけるテーブルであり、図２の３１０、３２０、３３０、３５０に対応する。図２の３１０、３２０、３３０、３５０は、システム２におけるインデックスＤＢ内の各テーブルを意味する。

【００３５】両者は構造的には全く同一であり、ユーザ情報に関するテーブル４５０は３５０と同じ内容を記録しておく。この入出力領域４００のＤＢに記録されるインデックスは、インデックスＤＢ３００（図１）の中の一部であり、蓄積されたインデックスに対応するイメージデータと合わせて可搬媒体５００に収まるデータ量であることが必要である。

【００３６】イメージ登録の際に、システムにより入出力領域４００へも登録すると判断されたときは、タイトルとＩＤは共にテーブル３１０と４１０に同一のデータを書き込む。これに対して、テーブル４２０は、テーブル３２０のデータを、転送位置切り替え部４９０を経由して、これにより変換して記録することで、可搬媒体５００におけるパス名を記録することとなる。４３０のレコードのデータ内容は、４１０と同様に、３３０と同一の構造のデータを、該当する部分だけ取り出したデータとして記録する。

【００３７】ファイルを可搬媒体５００によりシステム２の外に出す処理を、エクスポート処理と呼ぶ。エクスポートの際には、インポートと逆に、インデックスデータは既に媒体上に記録済みであり、イメージデータファイルをイメージＤＢ１００から可搬媒体５００へコピーすることが必要となる。

【００３８】登録の際に入出力領域４００にも書き込んだイメージファイルの識別情報は、既知の方法により移動することが可能である。例えば、インデックスＤＢ３００に識別テーブル３９０を設定し、そこに“ＭＯ１”と記載されているファイルを検索し、可搬媒体５００にコピーする方法が考えられる。

【００３９】以上の方法を応用することにより下記のようなシステムを構築することもできる。図４に示すのはデータ登録を専用に行うシステムの１つの構成例である。このシステムでは、登録されたデータを可搬媒体により他のシステムに移動して利用することを前提とするものである。そのため図１のシステムと比較して、継続的にインデックスデータを蓄積するインデックスＤＢ３００を廃し、入出力領域を４００に対応する、複数の入出力領域４００、６００、７００を有している。

【００４０】イメージの登録の際には、予め、その内容に合わせて４００、６００、７００のいずれにインデックスを記録するかを定め、インデックスＤＢ管理部２００は、登録するＤＢファイルに接続（アクセス）した状態で登録を行う。

【００４１】登録後、当該入出力領域（４００、６００、７００等の複数のもののうちいづれか１つ）と該当するイメージデータファイルを可搬媒体５００にコピーする。複数の入出力領域を切り替えながら登録する場合には、可搬媒体も複数用意することになる（可搬媒体５００、５１０）。その場合に上記のインデックスＤＢ３００に記録した識別テーブル３９０の媒体名称は“ＭＯ１”だけでなく複数の媒体を識別できるように媒体の名称若しくはこれに代る識別子を記録する。

【００４２】一方、複数の可搬媒体により他のシステムで登録されたデータを、読み取る側のシステムも上記と同様に、はじめからデータ登録は複数の可搬媒体により実行することを前提として構築することもできる。

【００４３】この場合、複数の入出力領域を持つことで、媒体単位で別々のＤＢにアクセスすることが最も簡単な構造である。１つのインデックスＤＢ３００に可搬媒体の中のインデックス情報をコピーすることで、システムを動作させることも、上記の方法を応用すれば可能である。

【００４４】登録用のシステム１でインデックスを作成する場合に、ユニークなデータであることを要求されるデータ、例えば、イメージＩＤ等は、利用する媒体毎に先頭文字を換えるなどして、重複を生じないようにする。こうすればデータをシステム２のインデックスＤＢ３００にコピーした場合に重複するＩＤは存在しない。そこで、各ＤＢファイルをテーブル単位に分割し、それを各テーブル毎に合成していくことにより、基本的に１つのＤＢとすることができる。ただし、このＤＢは機械的に接続しただけであり整合性が取られていない。整合性は、合成後にソート処理をすることにより解決できる。

【００４５】

【発明の効果】本発明によれば、ＲＤＢを用いたシステムにおいても、システム間でＤＢの一部のデータを分割して受け渡し機能させることを、容易に短時間で実行できる。

【図面の簡単な説明】

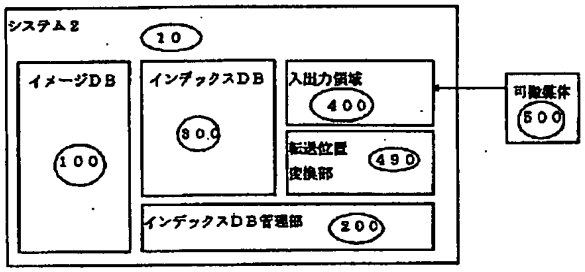
【図１】本システムの構成の一例を説明するための図である。

【図２】図１のシステムで用いるデータベースの構造を説明するための図である。

【図３】本システムで用いるデータベースにおいて、記載内容の切り替えを説明するための図である。
【図４】本発明の一応用例の構成を説明するための図である。
【符号の説明】
１００…イメージデータベース、２００…インデックスデータベース管理部、
３００…データベース本体であるインデックスデータベース、
４００…データベースの一部を別のシステムに移動するときに用いる入出力領域、
５００…可搬媒体、３１０…イメージタイトルのテーブル、
３２０…ファイルの格納位置を示すパス名を記録するテーブル、
３３０…各ファイル毎にアクセス可能なユーザを記録するセキュリティテーブル、
３５０…各ユーザを登録するユーザテーブル。

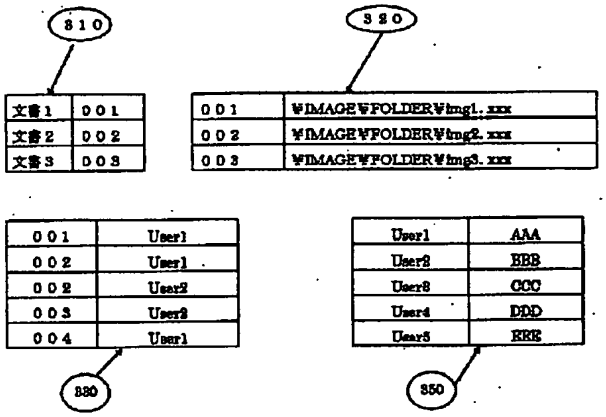
【図１】

図１



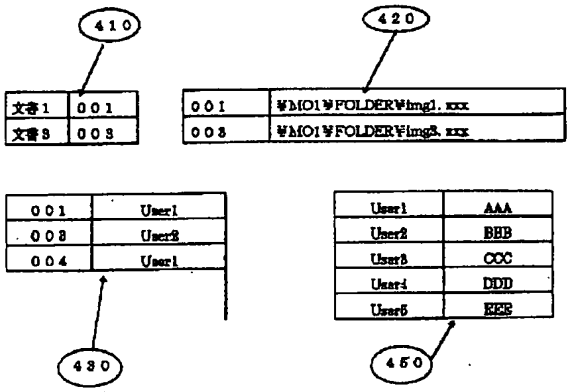
【図２】

図２



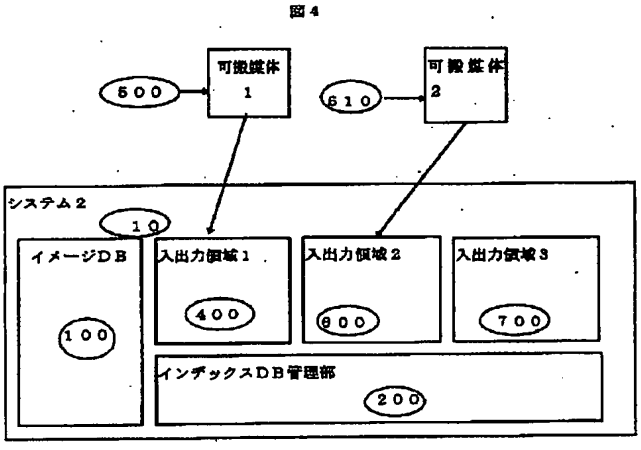
【図３】

図３



【図４】

図４



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号

特開平6－214849

(43)公開日　平成６年(1994)８月５日

(51)Int.Cl.[5]	識別記号	庁内整理番号	Ｆ Ｉ	技術表示箇所
Ｇ０６Ｆ　12/00	５２０　Ａ	8526－5B		
	５１２	8526－5B		

審査請求　未請求　請求項の数２　ＯＬ　（全 11 頁）

(21)出願番号　特願平5－4948

(22)出願日　平成５年(1993)１月14日

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地

(72)発明者　加藤　宣弘
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

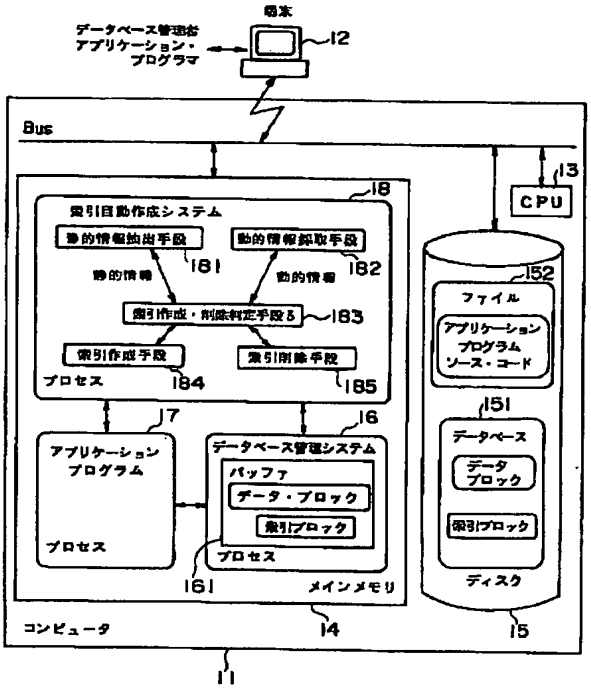
(74)代理人　弁理士　鈴江　武彦

(54)【発明の名称】　データベースシステム

(57)【要約】

【目的】高速なデータ操作を行なうための索引だけを自動作成し、データベース操作の高速化を図る。

【構成】索引自動作成システム１８は、アプリケーション・プログラム１７のデータ操作手続きの内容を抽出し、その抽出した内容とデータベース１５１のテーブルの構成情報とによって各データ操作手続きの実行に際して必要になる入出力データブロック数を求める。そして、そのブロック数に基づいて、索引作成の可否が索引作成・削除判定手段１８３により判断されて、データ操作の高速化に寄与する場合にのみ索引が自動作成される。したがって、無駄な索引がなくなり、常に高速なデータ操作を行なうことが可能になる。



【特許請求の範囲】
【請求項１】　各種アプリケーションプログラムからのデータ操作手続き要求に応じてリレーショナル・データベースのデータ操作を実行するデータベースシステムにおいて、
アプリケーション・プログラムのソース・コードに含まれるデータ操作手続きに関する情報を抽出する手段と、
この抽出したデータ操作手続きに関する情報とそのデータ操作手続きで扱われるテーブルの構成情報とに基づいて、データ操作手続きの実行のために２次記憶装置との間で入出力が必要なデータブロック数を、各データ操作手続き毎に算出する手段と、
この算出された各データ操作手続き毎の入出力データブロック数を統合し、その統合結果に従って、索引を作成するか否かを前記リレーショナル・データベースを構成する各テーブル毎に判定する手段と、
この判定結果に従って索引を作成する手段とを具備することを特徴とするデータベースシステム。
【請求項２】　データベースのデータにアクセスするためのデータ操作手続きを含むアプリケーションの実行が可能であり、索引を用いた検索が可能であるデータベースシステムにおいて、
システムを稼動する前に、アプリケーション・プログラムのソース・コードに含まれるデータ操作手続きの部分から静的な情報を抽出する静的情報抽出手段と、
システム稼動時にそれぞれのアプリケーションのデータ操作手続きの実行頻度、実行時間などの動的情報を採取する動的情報採取手段と、
上記静的情報抽出手段により注出された情報を基にそれぞれの表に対する索引を作成すべきどうかを、また上記静的情報抽出手段により注出された情報を基に表に対する索引を新たに作成するかどうか、あるいは既存の索引を削除するかどうかを判定する索引作成・削除判定手段と、
この判定結果にしたがって索引の作成、および作成された索引の削除を行なう索引作成・削除手段とを具備することを特徴とするデータベースシステム。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】この発明はデータベースシステムに関し、特に自動索引作成システムを持つリレーショナルデータベースシステムに関する。
【０００２】
【従来の技術】一般に、データベースのデータは、一定の大きさをそれぞれ持つ複数のデータ・ブロックから構成されている。これらデータ・ブロックは、磁気ディスクなどの外部記憶装置内に格納されている。
【０００３】リレーショナルデータベースにおいて、例えば、あるテーブルから特定のタプルを検索する場合には、検索対象のテーブルの先頭タプルを含むデータ・ブロックを見つけ、ディスクからメモリにそのブロックを読み出す。もしそのブロックに検索対象のタプルが含まれていれば、ディスクからの読み出しは終るが、そのブロックに含まれていなければ、次のブロックを読み出す。このようなデータ・ブロックの読み出し動作は、検索対象のタプルを含むブロックを読み出すまで続けられる。
【０００４】リレーショナルデータベースのテーブルは、多くのタプルから構成されている。データベースのデータを格納するデータ・ブロックの数は、タプルが多いほど増加される。データ・ブロックの数が少なければ、前述のような検索のためのディスクからメモリへのブロックの平均読み出し回数は少なくて済む。しかし、テーブルを構成するタプル数が多くそれらタプルが多くのブロックに格納されている場合には、ブロックの平均読み出し回数は多くなる。そこで、従来より、検索対象のタプルを含むブロックがどれであるかをより少ないブロック読み出し回数で知ることができるような手法が考えられている。
【０００５】その手法の１つが索引である。つまり、テーブルに含まれる各タプルの１つ以上のフィールドの値を用いてバランス木のようなデータ構造を作り、それを索引ブロックとして格納する。索引を用いたタプルの検索では、まずテーブルの先頭の索引ブロックをディスクから読み出し、検索対象のタプルのフィールドの値をキーとして、索引ブロック内を探索する。もし検索対象のタプルが格納されているデータ・ブロックの情報が、その索引ブロック内に存在すれば、その情報を用いてデータ・ブロックを読み出す。その索引ブロックに存在しなければ、索引データ構造に従って探索し、次の索引ブロックを読み出し、同様に検索対象のタプルが格納されているデータ・ブロックの情報が存在するかどうか調べる。索引ブロックの読み出しは、検索対象のタプルのデータ・ブロックの情報を見つけるまで行なう。
【０００６】一般に、索引ブロック数はデータ・ブロック数よりも少ない。また、索引ブロックの読み出し順はバランス木のようなデータ構造に基づいているので、一般に索引を用いたタプルの検索は、索引を用いない場合よりも、ブロックの読み出し回数が少なく、高速である。しかし、テーブルを構成するほとんど全てのタプルを検索するような場合には、ほとんど全てのデータ・ブロックを読み出す必要があるので、索引ブロックの読み出しが無駄になり、索引を用いない検索の方が高速になる。また、タプルの挿入、削除や、索引に使われているフィールド値の更新を頻繁に行なう場合には、対応する索引ブロックを頻繁に変更しなければならないので、索引がある場合にこれらの操作に多くの時間を要することになる。
【０００７】従来、システムで実行されるアプリケーションをそれぞれ高速化するために、データベース管理

者、アプリケーション・プログラマなどが、データベースの各テーブルに対して索引を作成していた。しかし、上記のように索引を作成することによりデータ操作が高速になる場合もあれば、そうでない場合もあるので、索引作成に関する知識を必要とした。また従来１つアプリケーション・プログラムを高速にするために、データ操作の実行トレースを採取できるシステムが存在した。しかし、一般にデータベースにアクセスするシステムでは、多数のアプリケーションが起動され、種々雑多なデータ操作が実行されるため、あるアプリケーションの実行を高速化するために作成した索引が、別のアプリケーション・プログラムの実行を遅くするという影響を及ぼすことがあった。

【０００８】

【発明が解決しようとする課題】従来のデータベースシステムでは、索引を作成することによりデータ操作が高速になる場合もあれば、逆に遅くなってしまうこともあり、索引を作成するか否かの判断には、データベース管理者等にはアプリケーション・プログラムの内容とデータベースの論理的・物理的構造に関する知識が必要された。

【０００９】また、多数のアプリケーションが起動されるシステムでは、あるアプリケーションの実行を高速化するために作成した索引が、別のアプリケーション・プログラムの実行を遅くするという影響を及ぼすことがあり、システム全体としての高い性能を達成するように索引を作成することは困難であった。

【００１０】この発明はこのような点に鑑みてなされたものであり、アプリケーション・プログラムおよびデータベースのテーブル内容に基づいて索引作成の可否を自動判定できるようにして、常に高速なデータ操作を行なうことができるデータベースシステムを提供することを目的とする。

【００１１】

【課題を解決するための手段および作用】この発明は、各種アプリケーションプログラムからのデータ操作手続き要求に応じてリレーショナル・データベースのデータ操作を実行するデータベースシステムにおいて、アプリケーション・プログラムのソース・コードに含まれるデータ操作手続きに関する情報を抽出する手段と、この抽出したデータ操作手続きに関する情報とそのデータ操作手続きで扱われるテーブルの構成情報とに基づいて、データ操作手続きの実行のために２次記憶装置との間で入出力が必要なデータブロック数を、各データ操作手続き毎に算出する手段と、この算出された各データ操作手続き毎の入出力データブロック数を統合し、その統合結果に従って、索引を作成するか否かを前記リレーショナル・データベースを構成する各テーブル毎に判定する手段と、この判定結果に従って索引を作成する手段とを具備することを第１の特徴とする。

【００１２】このデータベースシステムにおいては、アプリケーション・プログラムのデータ操作手続きの内容とテーブルの構成情報とによって各データ操作手続きの実行に際して必要になる入出力データブロック数が計算により求められ、そのブロック数に基づいて、索引作成の可否が判断されて、データ操作の高速化に寄与する場合にのみ索引が自動作成される。したがって、常に高速なデータ操作を行なうことが可能になる。

【００１３】また、この発明は、データベースのデータにアクセスするためのデータ操作手続きを含むアプリケーションの実行が可能であり、索引を用いた検索が可能であるデータベースシステムにおいて、システムを稼動する前に、アプリケーション・プログラムのソース・コードに含まれるデータ操作手続きの部分から静的な情報を抽出する静的情報抽出手段と、システム稼動時にそれぞれのアプリケーションのデータ操作手続きの実行頻度、実行時間などの動的情報を採取する動的情報採取手段と、上記静的情報抽出手段により注出された情報を基にそれぞれの表に対する索引を作成すべきどうかを、また上記静的情報抽出手段により注出された情報を基に表に対する索引を新たに作成するかどうか、あるいは既存の索引を削除するかどうかを判定する索引作成・削除判定手段と、この判定結果にしたがって索引の作成、および作成された索引の削除を行なう索引作成・削除手段とを具備することを第２の特徴とする。

【００１４】このデータベースシステムでは、システムを稼働する前にアプリケーションプログラムのデータ操作手続きとデータベースの静的な情報から、テーブルのフィールドに索引を作成するかどうかを判定し、有効な索引だけを作成する。またシステム稼動時の実行頻度、実行時間などの動的な情報から、既存の索引の有効性を検証し、また新たな索引を作成すべきかどうかを判定する。以上により、システム稼動時にも適宜既存の索引の有効性、新しい索引の必要性が判定されるので、システムに保持されるのは有効な索引だけになる。したがって、多数のアプリケーションが起動されるシステムでも、あるアプリケーションの実行を高速化するために作成した索引が、別のアプリケーション・プログラムの実行を遅くするという影響を及ぼすことがなくなり、システム全体としての高い性能を達成できる。

【００１５】

【実施例】以下、図面を参照してこの発明の実施例を説明する。図１には、この発明の一実施例に係わるデータベースシステムを実現するためのコンピュータシステムの構成が示されている。

【００１６】このデータベースシステムは、ホストコンピュータ１１およびこれに接続された端末１２を含むコンピュータシステムによって実現される。ホストコンピュータ１１は、ＣＰＵ１３、メインメモリ１４、および磁気ディスク装置１５を備えている。ＣＰＵ１３は、こ

のホストコンピュータ１１全体の動作を制御するためのものであり、メインメモリ１４に格納されたプログラムの実行によって、データベース処理に係わる種々の演算、および磁気ディスク装置１５の入出力制御を初め、索引自動作成のための処理等実行する。

【００１７】メインメモリ１４には、データベース管理システム１６、アプリケーションプログラム１７、および索引自動作成システム１８がそれぞれ実行対象のプロセスとして格納される。

【００１８】データベース管理システム１６は、ＳＱＬのようなデータ操作言語を用いて磁気ディスク装置１５のリレーショナルデータベース１５１の検索、更新等の各種のリレーショナルデータベース演算を実行するものであり、アプリケーションプログラム１７からの要求に従って動作される。また、データベース管理システム１６は、ビュー機能をサポートする。このビュー機能は、端末１２を操作するデータベース管理者に対して様々な視点のテーブルの視像を画面表示する機能である。バッファ１６１に読み出されたリレーショナルデータベース１５１のテーブルはその論理構造が異なるビューに変換され、それが端末１２に画面表示される。データベース管理者やアプリケーションプログラマは、画面表示されたビューを参照しながらデータベースの検索等を行なうことができる。

【００１９】アプリケーションプログラム１７は、データベース管理システム１６を利用した各種データベース操作を実行する。このアプリケーションプログラム１７は、実行時に磁気ディスク装置１５のファイル１５２からメインメモリ１４にロードされる。

【００２０】索引自動作成システム１８は、アプリケーション・プログラム１７およびデータベース１５１のテーブル内容に基づいて索引作成の可否を自動判定し、索引により高速なデータ操作を行なうことができる時には、索引を自動作成する。この索引自動作成システム１８は、静的情報抽出手段１８１、動的情報採取手段１８２、索引作成・削除判定手段１８３、索引作成手段１８４、および索引削除手段１８５から構成されている。

【００２１】静的情報抽出手段１８１は、システムを稼動する前、具体的には磁気ディスク装置１５のファイル１５２からメインメモリ１４にアプリケーション・プログラム１７をロードする時に、そのアプリケーションプログラムのデータ操作手続きの部分からデータ操作の種類、アクセスするテーブル、集約関数と集約関数を適用するフィールド、更新するフィールド、検索フィールドと検索条件を抽出する。

【００２２】動的情報採取手段１８２は、システム稼動時に、その時に実行されているアプリケーション・プログラム１７のデータ操作手続きの実行頻度、実行時間を採取する。

【００２３】索引作成・削除判定手段１８３は、静的情報抽出手段１８１により抽出された情報を基にそれぞれのテーブルに対する索引の候補が作成条件を満たすかどうかを判定する。また索引作成・削除判定手段１８３は、動的情報採取手段１８２により採取された情報を基に、それぞれのテーブルに対する索引を見直し、索引の作成、削除を判定する。索引作成手段１８４、および索引削除手段１８５は、索引作成・削除判定手段１８３の指示を受けて、それぞれ索引を作成、削除する。図２にテーブルの一例を示す。

【００２４】ここでは、データベース１５１に、従業員テーブル（ＥＭＰ）Ｔ１、所属テーブル（ＤＥＰＴ）Ｔ２、物品発注・納入テーブル（ＯＲＤＥＲ）Ｔ３が定義されている場合を考える。

【００２５】従業員テーブル（ＥＭＰ）Ｔ１は従業員に関する情報を保持するテーブルであり、所属テーブル（ＤＥＰＴ）Ｔ２は従業員の所属部所に関する情報を保持するテーブル、物品発注・納入テーブル（ＯＲＤＥＲ）Ｔ３は従業員が発注した物品の納入状況に関する情報を保持するテーブルである。

【００２６】いま、従業員テーブル（ＥＭＰ）Ｔ１には1000レコード、所属テーブル（ＤＥＰＴ）Ｔ２には10レコード、物品発注・納入テーブル（ＯＲＤＥＲ）Ｔ３には10000 レコードがそれぞれ含まれているとする。各フィールド名の右横に示した数は、フィールドのバイト数を表す。

【００２７】例えば、従業員テーブル（ＥＭＰ）Ｔ１においては、従業員番号（ＥＭＰ　ＮＯ）フィールドは４バイト、従業氏名（ＥＭＰ　ＮＡＭＥ）フィールドは２０バイト、従業員に関するその他の情報（ＥＭＰ　ＥＬＳＥ）のためのフィールドは１００バイト、従業員番号（ＤＥＰＴ　ＮＯ）は４バイトから構成されている。

【００２８】従って、従業員テーブル（ＥＭＰ）Ｔ１のレコードの大きさは１２８バイトとなる。同様に、所属テーブル（ＤＥＰＴ）Ｔ２のレコードの大きさは１２４バイト、物品発注・納入テーブル（ＯＲＤＥＲ）Ｔ３のレコードの大きさは１３２バイトとなる。図３にデータ・ブロックの例を示す。

【００２９】１つのデータ・ブロックのサイズは２０４８バイトである。ブロックの先頭の１００バイトはブロック・ヘッダとして確保され、最後の２００バイトは拡張領域として確保されている。残りの１７４８バイトがデータを格納するために用いられる。このデータ格納領域においては、各レコード毎に１０バイトのレコード・ヘッダが確保されている。従って、１レコード・サイズがＬバイトの場合には、１データ・ブロックに、１７４８／（Ｌ＋１０）の数のレコードが格納できることになる。ゆえに、従業員テーブル（ＥＭＰ）Ｔ１、所属テーブル（ＤＥＰＴ）Ｔ２、物品発注・納入テーブル（ＯＲＤＥＲ）Ｔ３のデータ・ブロックにはそれぞれ最大１２、１３、１２レコードが格納できる。

【００３０】いま従業員テーブル（ＥＭＰ）Ｔ１、所属テーブル（ＤＥＰＴ）Ｔ２、物品発注・納入テーブル（ＯＲＤＥＲ）Ｔ３にはそれぞれ１０００、１０、１００００レコードが含まれているので、それぞれ８４、１、８３４ブロックの領域を必要とする。図４に各テーブルに対するレコードあたりのバイト数、ブロックあたりのレコード数、ブロック数を示す。図５に索引ブロックの例を示す。

【００３１】索引ブロックにおいては、データ・ブロックと同様に、１７４８バイトが索引レコードを格納するために用いられる。索引を付けるフィールドの値とそれに対応するレコードの格納位置を示すポインタ（２０バイト）が、１つの索引レコードを形成する。例えば従業員テーブル（ＥＭＰ）Ｔ１のフィールド（ＥＭＰ＿ＮＯ）に索引を付けるとすると、１つの索引データに４＋２０＝２４バイト必要となり、１索引ブロックに７２索引レコードを格納できる。テーブルＴ１には１０００レコードがあるので、すべてのレコードに対する索引を作るためには、１４個の索引ブロックが必要となる。また、索引をバランス木で構成する場合には、バランス木の構成を格納するための索引構造ブロックが必要となる。索引構造ブロックには、索引ブロック数の約１割が必要であるので、上記の場合には１個の索引構造ブロックが必要となり、合わせて１５ブロックを索引のために確保する必要がある。

【００３２】図６に、各テーブルの主なフィールドに対する索引ブロック数、索引構造ブロック数を示す。ここで、所属テーブル（ＤＥＰＴ）Ｔ２のフィールドＤＥＰＴ＿ＮＯに対して索引構造ブロックがないのは、索引構造が索引ブロックに含まれているからである。

【００３３】図７には、図２のテーブルＴ１～Ｔ３に対して実行されるアプリケーション・プログラム１７の一例が示されている。図７（ａ）はある従業員が物品を発注するときのプログラム、（ｂ）は納入担当者が物品を納入するときのプログラム、（ｃ）はある部所全体の注文、納入の状況を表示するためのプログラムである。プログラムはデータ操作手続きの部分を中心に示す。プログラムにおいて英小文字はホスト言語とデータ操作言語とで共用される変数である。または[]は配列を表す。以下、それぞれのプログラムについて簡単に説明する。

【００３４】発注プログラムは、物品発注・納入テーブル（ＯＲＤＥＲ）Ｔ３の更新等を行なうためのものであり、この発注プログラムでは、まず入力された従業員番号と発注品名をそれぞemp noとorder nameにセットする。そして、物品発注・納入テーブル（ＯＲＤＥＲ）Ｔ３からフィールド（ＯＲＤＥＲ　ＮＯ）の最大値を検索し、それを１だけインクリメントしてorder　noにセットする。そして、テーブル（ＥＭＰ）Ｔ１からフィールドＥＭＰ　ＮＯの値がemp　noに等しいレコードを検索し、フィールド（ＤＥＰＴ　ＮＯ）の値をdept　noにセットする。最後に新しいレコードをテーブル（ＯＲＤＥＲ）Ｔ３に入力する。

【００３５】納入プログラムは、テーブル（ＯＲＤＥＲ）Ｔ３のフィールド（ＦＬＡＧ）を更新するためのものであり、この納入プログラムでは、入力された発注品番号（納入品番号）をorde　noにセットし、テーブル（ＯＲＤＥＲ）Ｔ３からフィールド（ＯＲＤＥＲ　ＮＯ）の値がdept　noに等しいレコードを検索し、そのフィールド（ＦＬＡＧ）の値を１に更新する。

【００３６】状況表示プログラムは、発注品名や納入状況等を画面表示するためのものであり、この状況表示プログラムでは、まず入力された部所番号をdept　noにセットし、テーブル（ＤＥＰＴ）Ｔ２からフィールドＤＥＰＴ　ＮＯの値がdept　noに等しいレコードを検索し、フィールドＤＥＰＴ　ＮＡＭＥの値をdept　nameにセットする。次に、テーブル（ＯＲＤＥＲ）Ｔ３からフィールドＤＥＰＴ　ＮＯの値がdept　noに等しいレコードを検索し、フィールドＯＲＤＥＲ　ＮＡＭＥ、ＦＬＡＧの値を配列order　name[]、flag[]にそれぞれセットする。ここで配列を用いるのは、この問合せでは複数のレコードを返される可能性があるからである。最後に、dept　name、order　name[]、flag[]を表示する。以上で説明したテーブルとアプリケーション・プログラムに対して本発明を適用した場合について以下説明する。

【００３７】静的情報抽出手段１８１は、図７に示すアプリケーション・プログラム１７から図８に示す情報を抽出する。図８には、それぞれのデータ操作手続きに対して、その識別子、手続きが含まれるアプリケーション・プログラム、その命令の種類、アクセスするテーブル名、値を更新するフィールド名、利用する集約関数、集約関数を適用するフィールド名、検索に使われるフィールド名、検索条件が含まれる。図８において、横線が引かれている箇所は対応する項目がないことを表す。更新フィールド名の＊はすべてのフィールドを更新することを表す。検索条件の＝ｖａｌは、ある値に等しい値を持つフィールドを含むレコードだけを選択することを表す。

【００３８】システムを稼動する前に索引作成・削除判定手段１８３は、図８に示す情報から各データ操作手続きに対するレコードの選択率を次のように計算する。ここで選択率とは、データ操作手続きにより選択したレコード数の全レコード数に占める割合である。

【００３９】図９に各データ操作手続きに対する選択率を示す。ただし、レコードを選択しないデータ操作手続きに対しては横線を引いておく。レコードの選択率は次のように求める。例えば、テーブル（ＥＭＰ）Ｔ１のフィールドＤＥＰＴ　ＮＯに対するデータ操作手続き２の選択率は次のようにして求めることができる。１０００人の従業員の中から特定の従業員１人を選択するから、選択率は１／１０００＝０．００１となる。

【００４０】さらに、索引作成・削除判定手段１８３

は、図４、図６、図８、図９に示す情報から、索引を作成しない場合のブロック読み出し数と索引を作成する場合のブロック読み出し回数を計算する。例えば、データ操作手続き１に対して、索引を作成しない場合、および索引を作成する場合それぞれのブロック読み出し回数を求める。索引を作成しない場合にはテーブル（ＯＲＤＥＲ）Ｔ３のすべてのデータ・ブロックを読まなければならないので、８３４ブロックとなる。一方、索引を作成する場合には、索引構成ブロックを１４ブロック、索引ブロックを１ブロック、データ・ブロックを１ブロック、合計１６ブロックだけ読めばよい。したがって、テーブル（ＯＲＤＥＲ）Ｔ３のフィールドＥＭＰ　ＮＯに対する索引を作成すると仮判定する。

【００４１】また、データ操作手続き６に対して同様に計算してみると、索引を作成しない場合には、すべてのデータ・ブロックを読まなければならないので、８３４ブロックとなる。一方、索引を作成する場合には次のように計算できる。

【００４２】テーブル（ＯＲＤＥＲ）Ｔ３は１００００レコードを含んでおり、データ操作手続き６の選択率は０．１なので、１０００レコードを選択することになる。索引ブロックにおいて、索引レコードはＤＥＰＴ　ＮＯの値の順にソートされているので、あるＤＥＰＴ　ＮＯの値の索引レコードは１４ブロックに保持されている。データ・ブロックにおいて、レコードはランダムに配置されているので、あるＤＥＰＴ　ＮＯの値に対する１００レコードは５８３ブロックに配置されている。ゆえに、索引を作成する場合には、索引構造ブロックを１４ブロック、索引ブロックを１４ブロック、データ・ブロックを５８３ブロック、合計６１１ブロック読まなければならない。したがって、テーブル（ＯＲＤＥＲ）Ｔ３のフィールドＤＥＰＴ　ＮＯに対する索引を作成すると仮判定する。

【００４３】図１０にそれぞれのデータ操作手続きに対して、索引を作成しない場合、作成する場合の読み出しブロック数、索引を作成するか否かを示す。データ操作手続き３は、テーブル（ＯＲＤＥＲ）Ｔ３のすべてのフィールドを更新するので、それらのフィールドに対して索引を作成しないと仮判定する。データ操作手続き３（ＩＮＳＥＲＴ命令）に対しては、入出力（読み出しと書き込み）ブロック数を示す。索引がない場合には空き領域のあるブロックを読み書きするので、入出力ブロック数は２となる。索引がある場合には、それに加えて索引ブロック、索引構造ブロックの修正のために入出力ブロック数が多くなる。テーブル（ＯＲＤＥＲ）Ｔ３にはフィールドＯＲＤＥＲ　ＮＯとＤＥＰＴ　ＮＯに索引を作成する可能性があるので、最大索引構造ブロックを１４ブロック、索引ブロックを２ブロックを読み出し、書き込むことになる。したがって、合計２＋（１４＋２）＊２＝３４となる。

【００４４】索引作成・削除判定手段１８３は、図８、図１０からテーブルのフィールドに索引を作成するかどうかを判定する。テーブル（ＯＲＤＥＲ）Ｔ３のフィールドＯＲＤＥＲ　ＮＯ、ＤＥＰＴ　ＮＯに対しては、索引を作成しないという仮判定と索引を作成するという仮判定がある。各データ操作手続きが同じ頻度で実行されるならば、図１０よりフィールドＯＲＤＥＲ　ＮＯ、ＤＥＰＴ　ＮＯのそれぞれにおいて、索引を作成する場合は索引を作成しない場合よりも入出力ブロック数が少ない。ゆえに、それぞれ索引を作成すると判定する。テーブルＥＭＰのフィールドＥＭＰ　ＮＯには索引を作成すると判定し、テーブルＤＥＰＴのフィールドＤＥＰＴ　ＮＯには索引を作成しないと判定する。索引作成手段１８４は、これらの判定に基づいて索引を作成する。なお、静的情報による索引の作成判定において、入出力ブロック数だけでなく、各種オーバヘッドの見積りを含めてもよい。

【００４５】動的情報採取手段１８２は、システム稼動時に、実行されるアプリケーションプログラムによる実際のデータ操作手続きの実行頻度、平均実行時間を採取し、記憶する。いま図１１に示すような実行頻度と平均処理時間が得られたとする。

【００４６】索引作成・削除判定手段１８３は、各データ操作手続きに対して、図１１に示す実行頻度を図１０の入出力ブロック数に乗じる。その結果を図１２に示す。図１２に示す値は各データ操作手続きが１秒あたりに入出力するブロック数を表す。この値を用いて、上記と同じように索引を作成するか否かの判定を行なう。

【００４７】テーブル（ＯＲＤＥＲ）Ｔ３のフィールドＯＲＤＥＲ　ＮＯに索引を作成するか否かで影響を受けるのは、データ操作手続き１、３、４、である。索引を作成しない場合には、それらの入出力ブロック数の和が１１８６４になるが、作成する場合には５７７となる。したがって、テーブル（ＯＲＤＥＲ）Ｔ３のフィールドＯＲＤＥＲ　ＮＯには索引を作成すると判定する。テーブル（ＯＲＤＥＲ）Ｔ３のフィールドＤＥＰＴ　ＮＯに索引を作成するか否かで影響を受けるのは、データ操作手続き３と６である。索引を作成しない場合には、それらの入出力ブロック数の和が１０２２になるが、作成する場合には１０８３となる。したがって、テーブル（ＯＲＤＥＲ）Ｔ３のフィールドＤＥＰＴ　ＮＯには索引を作成しないと判定する。この判定は、静的情報から求めた上記の判定とは異なるので、索引削除手段１８５に対して既に作成されている索引の削除を指示する。

【００４８】索引を見直した後、動的情報採取手段１８２は、データ操作手続きの実行頻度、平均実行時間を採取し、記憶する。索引作成・削除判定手段１８３は、実行頻度と平均実行時間との積のデータ操作手続きに対する総和を、索引の見直しの前後で比較する。もし見直し後の値の方が大きければ、索引を見直し以前の状態に戻

す。

【００４９】なお、各データ操作手続きに対する入出力ブロック数は、静的情報から得られた値ではなく、動的情報から得られた値を採用してもよい。その場合には、動的情報採取手段１８２は、システム稼動時にデータ操作手続きの平均入出力ブロック数も採取する。また、索引の見直に入出力ブロック数ではなく、平均実行時間を用いることもできる。そのためには、索引がある場合とない場合についてデータ操作手続きの実行時間を採取する必要がある。次に、図１３のフローチャートを参照して、索引作成の手順を説明する。

【００５０】まず、静的情報抽出手段１８１は、実行対象のアプリケーションプログラムのデータ操作手続きの部分から静的情報を抽出する（ステップＳ１１）。次いで、索引作成・削除判定手段１８３は、静的情報とデータ操作手続きの選択率に基づき、各データ操作手続きの入出力ブロック数を計算する（ステップＳ１２）。そして、索引作成・削除判定手段１８３は、各データ操作手続きの入出力ブロック数に基づき、索引を作成するか否かを仮判定する（ステップＳ１３）。

【００５１】この後、索引作成・削除判定手段１８３は、索引作成の仮判定結果を統合し索引を作成するかどうかを判定する（ステップＳ１４）。そして、索引作成手段１８４は、判定結果に従って、必要なテーブルの索引を作成する。以上、ステップＳ１１～Ｓ１５の処理はアプリケーションプログラムの実行前に行なわれる処理である。この後、アプリケーションプログラムの実行時には、次のステップＳ１６～Ｓ１８の処理が繰り返し実行される。

【００５２】動的情報採取手段１８２は、各データ操作手続きの実行頻度、実行時間を測定する（ステップＳ１６）。次いで、索引作成・削除判定手段１８３は、各データ操作手続きの実行頻度、実行時間からなる動的情報に基づき、索引の見直しを行なう（ステップＳ１７）。そして、索引作成手段１８４および索引削除手段１８５は、それぞれ見直し結果に基づいて、索引の作成・削除を行なう（ステップＳ１８）。

【００５３】以上のように、この実施例においては、アプリケーション・プログラム１７のデータ操作手続きの内容とテーブルの構成情報とによって各データ操作手続きの実行に際して必要になる入出力データブロック数が計算により求められ、そのブロック数に基づいて、索引作成の可否が判断されて、データ操作の高速化に寄与する場合にのみ索引が自動作成される。したがって、常に高速なデータ操作を行なうことが可能になる。

【００５４】また、システム稼動時にも適宜既存の索引の有効性、新しい索引の必要性が判定されるので、システムに保持されるのは有効な索引だけになる。したがって、多数のアプリケーションが起動されるシステムでも、あるアプリケーションの実行を高速化するために作成した索引が、別のアプリケーション・プログラムの実行を遅くするという影響を及ぼすことがなくなり、システム全体としての高い性能を達成できる。尚、この発明は、ここに示した実施例に限定されるものではなく、その請求範囲を逸脱しない範囲で適宜変更可能である。

【００５５】

【発明の効果】以上のようにこの発明によれば、データベース管理者の手を煩わせることなく、テーブルに有効な索引を作成できる。しかも、アプリケーション・プログラム全体を見通して索引作成の可否を判定するので、システム全体の性能向上を見込める。また、システム稼動時にも適宜既存の索引の有効性、新しい索引の必要性を判定するので、システムに保持されるのは有効な索引だけに淘汰されていく。

【図面の簡単な説明】

【図１】この発明の一実施例に係るデータベースシステムの構成を示すブロック図。

【図２】同実施例のデータベースシステムで扱われるデータベースのテーブルの一例を示す図。

【図３】同実施例のデータベースシステムによって入出力されるデータ・ブロックの一例を示す図。

【図４】図３に示した各テーブルに対するレコードあたりのバイト数、レコード数、ブロックあたりのレコード数、ブロック数を示す図。

【図５】同実施例のデータベースシステムによって入出力される索引ブロックの一例を示す図。

【図６】図３に示した各テーブルの主なフィールドに対する索引ブロック数、索引構造ブロック数を示す図。

【図７】同実施例のデータベースシステムで実行されるアプリケーション・プログラムの一例を示す図。

【図８】図７のアプリケーション・プログラムのデータ操作手続きから得られる静的情報の一例を示す図。

【図９】図７のアプリケーション・プログラムの各データ操作手続きによるレコード選択率を示す図。

【図１０】図７のアプリケーション・プログラムの各データ操作手続きに対する入出力ブロック数と仮判定結果を示す図。

【図１１】同実施例のデータベースシステムで採取される動的情報の例を示す図。

【図１２】同実施例のデータベースシステムで採取される動的情報を考慮した場合の各データ操作手続きに対する入出力ブロック数を示す図。

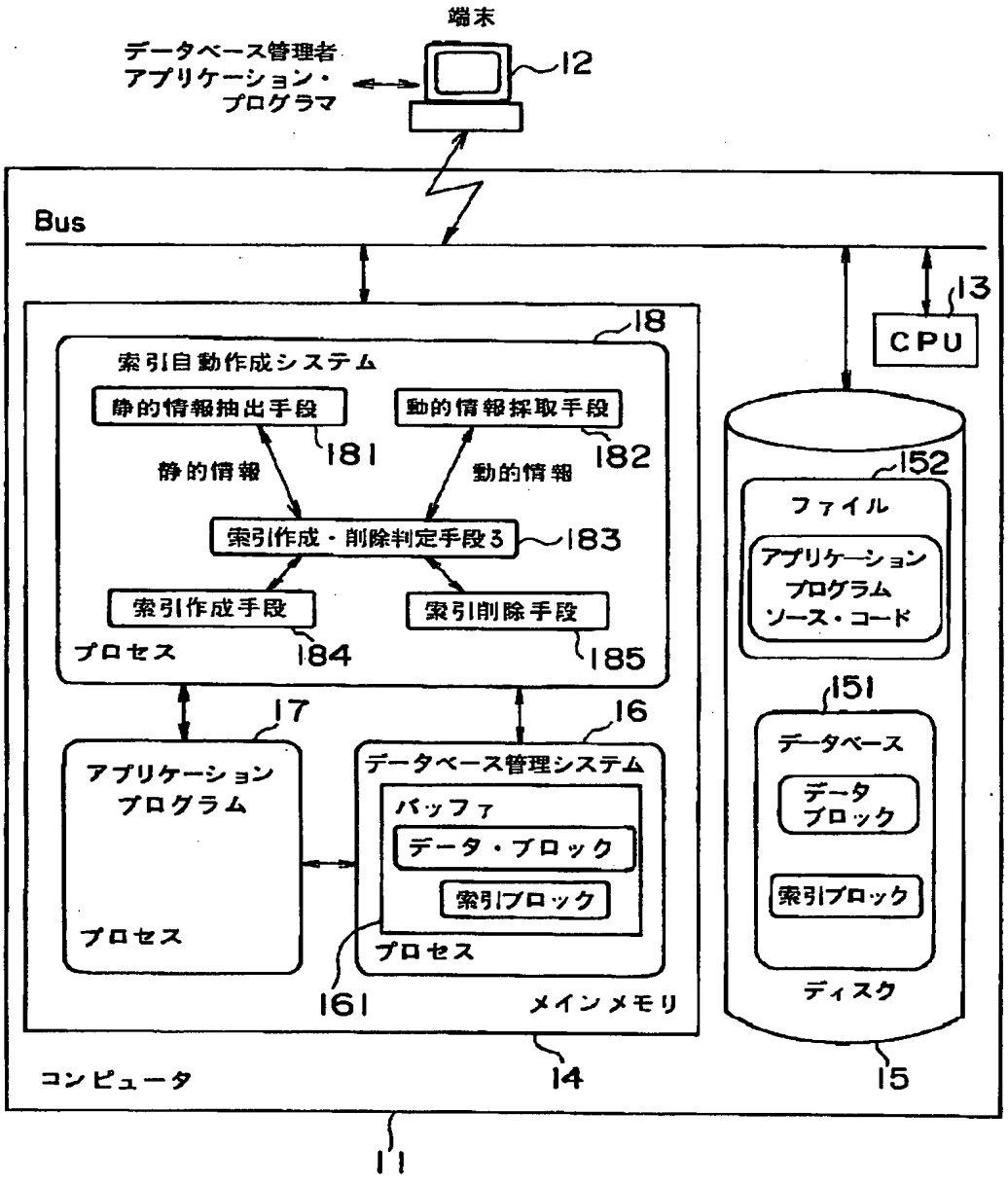
【図１３】同実施例のデータベースシステムにおける索引自動作成処理を説明するフローチャート。

【符号の説明】

１１…ホストコンピュータ、１２…端末、１３…ＣＰＵ、１４…メインメモリ、１５…磁気ディスク装置、１６…データベース管理システム、１７…アプリケーションプログラム、１８…索引自動作成システム、１８１…静的情報抽出手段、１８２…動的情報採取手段、１８３

…索引作成・削除判定手段、１８４…索引作成手段、１　　８５…索引削除手段。

【図１】



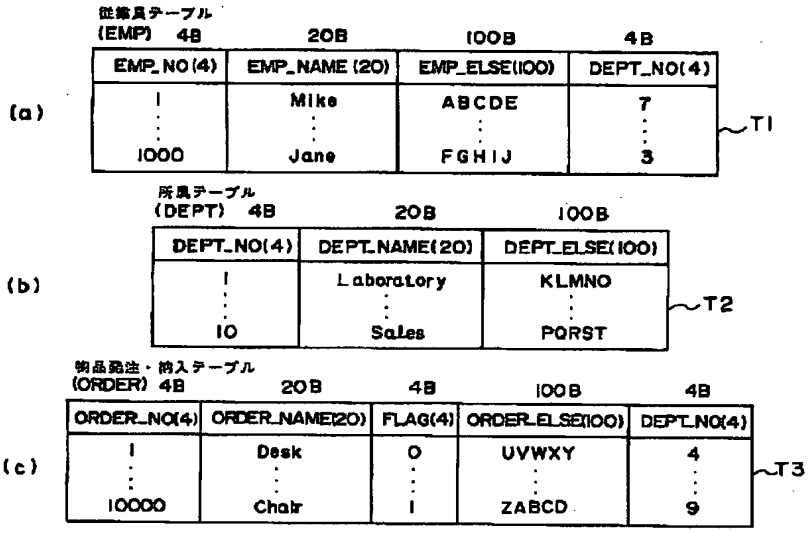
【図４】

テーブル名	レコードあたりのバイト数	ブロックあたりのレコード数	レコード数	ブロック数
EMP	138	12	1000	84
DEPT	134	13	10	1
ORDER	142	12	10000	834

【図６】

テーブル	フィールド	索引レコードあたりのバイト数	ブロックあたりのレコード数	レコード数	索引構造ブロック数	索引ブロック数
EMP	EMP_NO	4+20	72	1000	14	1
EMP	DEPT_NO	4+20	72	1000	14	1
DEPT	DEPT_NO	4+20	72	10	1	0
ORDER	ORDER_NO	4+20	72	10000	139	14
ORDER	DEPT_NO	4+20	72	10000	139	14

【図２】



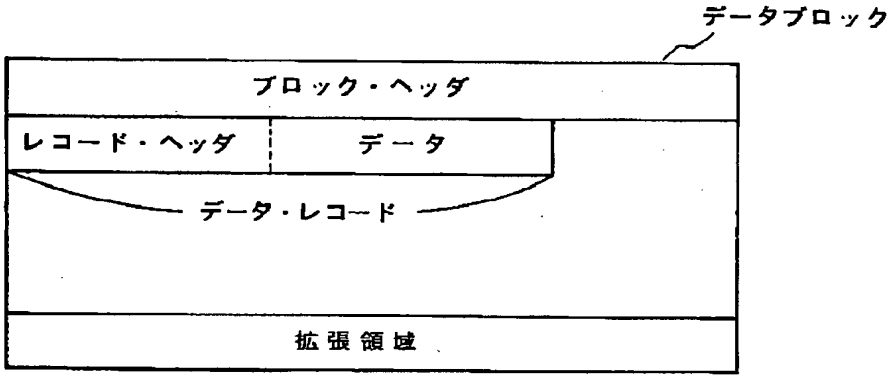
【図９】

データ操作手続き識別子	選択率
1	0.0001
2	0.001
3	―――
4	0.0001
5	0.1
6	0.1

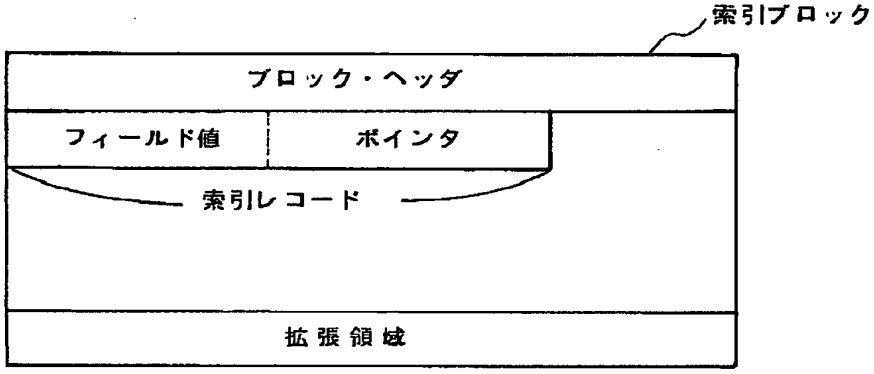
【図１１】

データ操作手続き識別子	実行頻度 (/sec)	実行時間 (sec)
1	10.3	0.21
2	10.3	0.15
3	10.3	1.20
4	3.9	0.27
5	1.2	0.13
6	1.2	3.78

【図３】



【図５】



【図７】

発注

```
INPUT emp_no, order_name

SELECT MAX(ORDER_NO)+1
INTO order_no
FROM ORDER;

SELECT DEPT_NO
INTO dept_no
FROM EMP
WHERE EMP_NO= emp_no;

INSERT INTO ORDER
VALUES (order_no, order_name,
        0, dept_no);
```

(a)

表示

```
INPUT dept_no

SELECT DEPT_NAME
INTO dept_name
FROM DEPT
WHERE DEPT_NO= dept_no;
SELECT ORDER_NAME, FLAG
INTO order_name[], flag[]
FROM ORDER
WHERE DEPT_NO= dept_no;

DISPLAY dept_name, order_name[], flag[]
```

(c)

納入

```
INPUT order_no

UPDATE ORDER
SET FLAG=1
WHERE ORDER_NO=order_no;
```

(b)

【図１２】

データ操作手続き識別子	入出力ブロック数（／ｓｅｃ）	
	索引がない場合	索引がある場合
1	8590	165
2	865	31
3	21	350
4	3253	62
5	1	2
6	1001	733

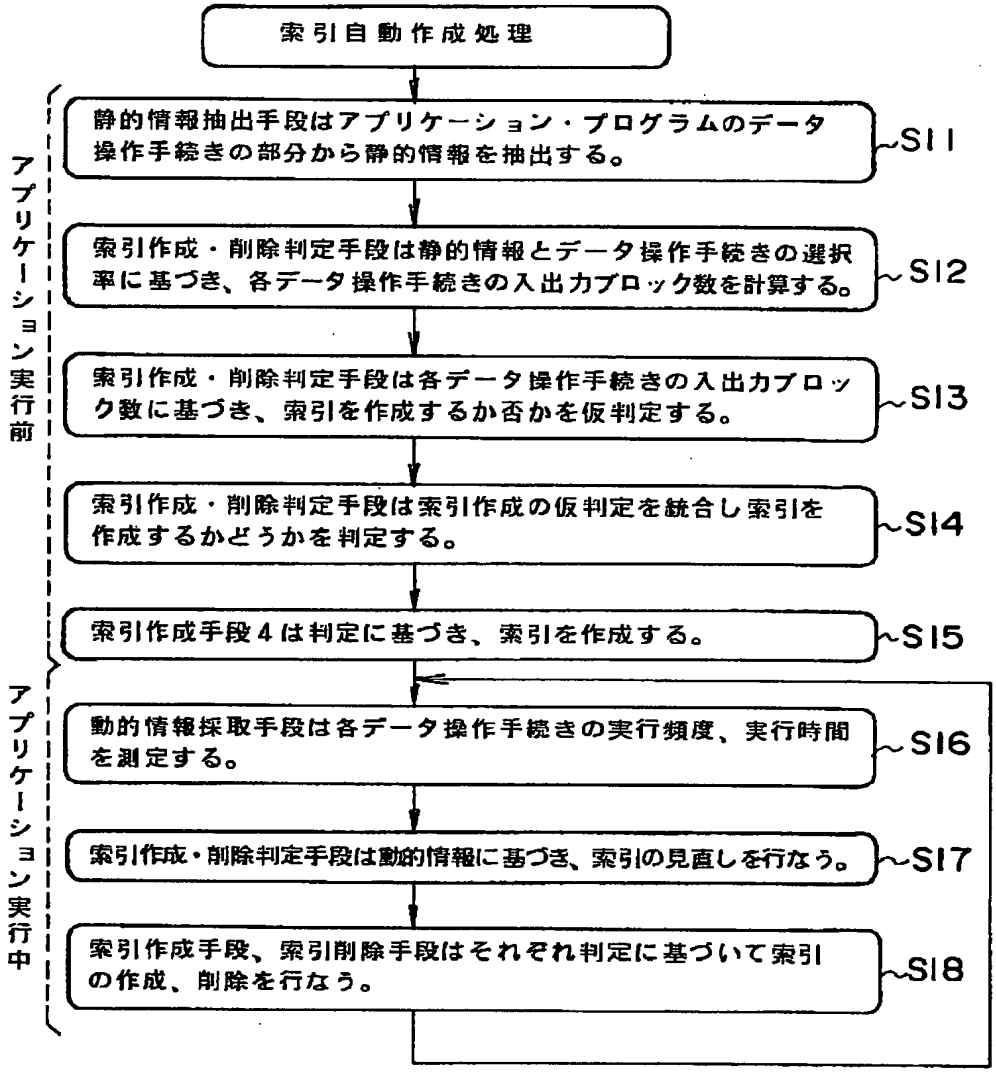
【図８】

データ操作手続き識別子	プログラム	コマンド	テーブル	集約関数	集約関数適用フィールド	更新フィールド	探索フィールド	探索条件
1	発注	SELECT	ORDER	MAX	ORDER_NO	———	———	———
2	発注	SELECT	EMP	———	———	———	EMP_NO	=val
3	発注	INSERT	ORDER	———	———	*	———	———
4	納入	UPDATA	ORDER	———	———	FLAG	ORDER_NO	=val
5	表示	SELECT	DEPT	———	———	———	DEPT_NO	=val
6	表示	SELECT	ORDER	———	———	———	DEPT_NO	=val

【図１０】

データ操作手続き識別子	入出力ブロック数		仮判定
	索引がない場合	索引がある場合	
1	834	16	作成する
2	84	3	作成する
3	2	34	作成しない
4	834	16	作成する
5	1	2	作成しない
6	834	611	作成する

【図１３】


索引自動作成処理
アプリケーション実行前
静的情報抽出手段はアプリケーション・プログラムのデータ操作手続きの部分から静的情報を抽出する。
S11
索引作成・削除判定手段は静的情報とデータ操作手続きの選択率に基づき、各データ操作手続きの入出力ブロック数を計算する。
S12
索引作成・削除判定手段は各データ操作手続きの入出力ブロック数に基づき、索引を作成するか否かを仮判定する。
S13
索引作成・削除判定手段は索引作成の仮判定を統合し索引を作成するかどうかを判定する。
S14
索引作成手段４は判定に基づき、索引を作成する。
S15
アプリケーション実行中
動的情報採取手段は各データ操作手続きの実行頻度、実行時間を測定する。
S16
索引作成・削除判定手段は動的情報に基づき、索引の見直しを行なう。
S17
索引作成手段、索引削除手段はそれぞれ判定に基づいて索引の作成、削除を行なう。
S18